CALIFORNIA INSTITUTE OF TECHNOLOGY
Computer Science Department

MEMO 2069

Hierarchical Power Routing
(21 figures)

by

Dave Johannsen

October 17, 1978

Advances in LSI technology allow the system designer to implement large amounts of processing capability on a single silicon chip. It will soon be possible to construct a large number of processing elements on these chips. How will the system designer organize these processing elements? Hierarchically designed array or tree machines are two possible alternatives. This paper provides a background for study of array and tree machines by examining how to supply power to an array of processing elements.

We will organize the processing elements into a heirarchy of *modules*, with each module containing α^2 *submodules*. α is the *branching ratio* of the physical design: there are α submodules supplied by one VDD or Ground line, and α of these groups of submodules are supplied by larger VDD and Ground lines. The processing elements are the submodules of the lowest level module. There are a total of $S=\alpha^{2n}$ processing elements in the entire array, and the heirarchy is n levels deep.

By examining the necessary sizes of wire needed to supply power to each submodule, we can approximate the area of each module, and the area of the entire processor array. The width of each wire is proportional to the current that flows through the wire [3], which in turn is proportional to the number of processing elements that receive power from the wire. We define the unit wire width β to equal the current I_{cell} needed by one processing element times the design rule ω which specifies the width of wire per unit current:

$$\beta = \omega I_{cell} \qquad (1)$$

All dimensions will be normalized to the size of one processing element, so ω is in units of cell size/unit current.

Square Processing Elements

We will examine four power routing schemes on an array of square processing elements: Tree, Comb, Staggered Tree, and Staggered Comb. Figures 1-4 illustrate each of these four schemes.

Both Tree structure power routing schemes supply each module with VDD from the center of one side and Ground from the center of the opposite side. The VDD line runs horizontally through the center of the module with vertical lines branching off to supply VDD to the submodules. The Ground line is routed around the perimeter of the module, again with vertical wires supplying the submodules.

Both Comb structure power routing schemes supply VDD to an upper corner of the module and Ground from the diametrically opposite corner. The VDD line runs horizontally across the top of the module with vertical lines branching off to supply the submodules. The Ground line runs along the bottom of the module, again with vertical lines supplying the submodules.

In the Staggered Tree and Staggered Comb cases, the line widths taper as the power demand through the wire decreases, while in the Tree and Comb cases, the width of each line is constant throughout its length.

We can tile the plane with processing elements with the above schemes, as can be seen by the following arguments which illustrate the Tree case:

Assume that a module of level i-1 can route VDD and Ground to each of its α^{2i-2} processing elements using the method described above, and that the top level of the resulting array is of the form shown in figure 1. By arranging α^2 of these submodules as shown in figure 1, mirroring those submodules where the VDD line enters from the right, we can route VDD and Ground to these α^2 submodules and produce a module of level i that can route VDD and Ground to each of its $\alpha^2\alpha^{2(i-1)}=\alpha^{2i}$ processing elements. By a similar argument it can be shown that level 1 can route VDD and Ground to α^2 processing elements. This inductively proves that the proposed power routing schemes can supply power to $S=\alpha^{2n}$ processing elements, for any n.

The proposed schemes will be compared in terms of size, area, signal path length, and other quantities associated with the array of processing elements.

Size Calculations

The first comparison will be in terms of array size. Since the proposals are recursively defined, the height and width of the modules can be expressed by recurrence relations. In the case of the tree, the height Y_{tree} has the following relation:

$$Y_{tree_0} = 1$$
$$Y_{tree_i} = \alpha Y_{tree_{i-1}} + \beta\alpha^{2i} + (\alpha-1)\beta\alpha^{2i-1} \quad (2)$$

The first term in the Y_{tree_i} equation accounts for the size of the α submodules, the second term accounts for the VDD line in the center of the module, and the third term accounts for the Ground lines at the top and bottom of the module. The solution to the above relation is

$$Y_{tree_i} = \alpha^i [\, 1 - (2\alpha-1)\beta/(\alpha-1) + (2\alpha-1)\beta\alpha^i/(\alpha-1) \,] \quad (3)$$

In all practical designs, $\beta \ll 1$, so we can make an approximation to (3):

$$Y_{tree_i} \approx \alpha^i [\, 1 + (2\alpha-1)\beta\alpha^i/(\alpha-1) \,] \quad (4)$$
$$Y_{tree_i} \approx \alpha^i (1 + \gamma_{tree}\beta\alpha^i) \quad (5)$$

where

$$\gamma_{tree} = (2\alpha-1)/(\alpha-1) \quad (6)$$

In a similar manner, we have the following relations for the width X_{tree}:

$$X_{tree_0} = 1$$

$$X_{tree_i} = \alpha X_{tree_{i-1}} + \beta\alpha^{2i}/2 + (2\alpha-1)\beta\alpha^{2i-1}/2 \quad (7)$$

$$X_{tree_i} = \alpha^i [\, 1 - (3\alpha-1)\beta/2(\alpha-1) + (3\alpha-1)\beta\alpha^i/2(\alpha-1) \,] \quad (8)$$

$$X_{tree_i} \approx \alpha^i [\, 1 + (3\alpha-1)\beta\alpha^i/2(\alpha-1) \,] \quad (9)$$

$$X_{tree_i} \approx \alpha^i (1 + \delta_{tree}\beta\alpha^i) \quad (10)$$

where

$$\delta_{tree} = (3\alpha-1)/2(\alpha-1) \quad (11)$$

The first term in equations (5) and (10) accounts for the processing elements themselves, while the second term accounts for the sum of the widths of the power lines. γ_{tree} and δ_{tree} are thus an efficiency measure of the Tree structure power routing scheme. The minimum value for γ and δ for any power scheme is 1, since each module must have a wire of width $\beta\alpha^{2i}$ supplying the power. Bounds for the array dimensions Y_{tree} and X_{tree} can be determined by evaluating γ_{tree} and δ_{tree} with $\alpha=2$ and in the limit as $\alpha\to\infty$.

$$\gamma_{tree}|_{\alpha=2} = 3 \quad (12)$$

$$\delta_{tree}|_{\alpha=2} = 5/2 \quad (13)$$

$$\gamma_{tree}|_{\alpha\to\infty} = 2 \quad (14)$$

$$\delta_{tree}|_{\alpha\to\infty} = 3/2 \quad (15)$$

This shows that the Tree structure is not very efficient: the total height of all the power lines is 2-3 times larger than the minimum wire height needed by the module, and the total width of all the power lines is 3/2 - 5/2 times larger than the minimum necessary. The corresponding formulas for the other power routing proposals are of the same form as (5) and (10). The results are tabulated here:

	Tree	Comb	Stree	Scomb
γ	$(2\alpha-1)/(\alpha-1)$	$(2\alpha-1)/(\alpha-1)$	1	1
$\gamma\|_{\alpha=2}$	3	3	1	1
$\gamma\|_{\alpha\to\infty}$	2	2	1	1
δ	$(3\alpha-1)/2(\alpha-1)$	$(2\alpha-1)/(\alpha-1)$	$(2\alpha+1)/2(\alpha-1)$	1
$\delta\|_{\alpha=2}$	5/2	3	5/2	1
$\delta\|_{\alpha\to\infty}$	3/2	2	1	1

It can be seen from the table that the Staggered Comb structure actually attains the optimal efficiency in wire usage, and that the array size is independant of the branching ratio used. This second fact may be useful when designing arrays with communication trees because the branching ratio of the power lines can be made the same as the branching ratio of the communication tree, which simplifies the design of the array.

The area A_i for the array is the product of the X_i and Y_i dimensions. Since both X_i and Y_i are quadratic in α^i, A_i is of degree 4 in α^i. The number of processors in the entire array is $\alpha^{2n}=S$, so the area of the entire array A_n is quadratic in S! This fact may limit the useful size of the array, because as S grows, an increasing percentage of the array area is used just to run the power lines. The following equations give the exact and approximate areas for the modules:

$$A_i = \alpha^{2i} [\, 1 - \delta\beta + \delta\beta\alpha^i \,][\, 1 - \gamma\beta + \gamma\beta\alpha^i \,] \tag{16}$$

$$A_i \approx \alpha^{2i} (1 + \delta\beta\alpha^i) (1 + \gamma\beta\alpha^i) \tag{17}$$

$$A_n \approx S (1 + \delta\beta S^{1/2}) (1 + \gamma\beta S^{1/2}) \tag{18}$$

Using the area approximation for the entire array (18), we can determine the ratio of total area to processing element area R, which is expressed in terms of $W=\beta S^{1/2}$.

$$R \approx 1 + (\gamma+\delta)W + \gamma\delta W^2 \tag{19}$$

R is plotted for each of the power routing schemes in figure 5. It is interesting to note that W is the width of the wire than would be required to supply power to one row of the final array.

R-1 gives the ratio of area taken by the power lines to area of a processing element. For practical designs, this value should be kept small. To minimize R, the staggered comb structure should be used, and β should be minimized, requiring the use of low-power processing elements. The fact that arrays constructed of low-power processing elements have less area than similar arrays built with higher-power processing elements can lead to an interesting result: a power/area design decision where lower cell power is chosen over smaller cell area will not only use less power, *but may also have a smaller total area!* The following table lists the approximate number of processing elements S that would give R=2 and R=4:

	Tree	Comb	Stree	Scomb
$R=2\|_{\alpha=2}$	$.151\beta^{-2}$	$.138\beta^{-2}$	$.243\beta^{-2}$	$.414\beta^{-2}$
$R=2\|_{\alpha\to\infty}$	$.237\beta^{-2}$	$.309\beta^{-2}$	$.414\beta^{-2}$	$.414\beta^{-2}$
$R=4\|_{\alpha=2}$	$.364\beta^{-2}$	$.333\beta^{-2}$	$.600\beta^{-2}$	$1.00\beta^{-2}$
$R=4\|_{\alpha\to\infty}$	$.574\beta^{-2}$	$.651\beta^{-2}$	$1.00\beta^{-2}$	$1.00\beta^{-2}$

Wire Contributions

We can determine where most of the power line area comes from by examining the area contributions from the wires of each level in the hierarchy. If the majority of the area is taken by power wires in either the highest level or the lowest level in the hierarchy, it may be advantageous to use an alternate power routing scheme or α for that level. If each level contributes approximately the same amount of area, alternate routing schemes would not benefit the total area greatly.

The power wire area for each module, D_i, can be determined by subtracting the area of the α^2 submodules from the total module area. Hence,

$$D_i = A_i - \alpha^2 A_{i-1} \tag{20}$$

$$D_i = [\ \gamma + \delta - 2\gamma\delta\beta + \gamma\delta\beta\alpha^i(\alpha+1)/\alpha\ ]\ \alpha^{3i}\beta(\alpha-1)/\alpha \tag{21}$$

Each module of level i is included in the complete array α^{2n-2i} times, so the total area contribution C_i of the power wires from level i is

$$C_i = \alpha^{2n-2i} D_i \tag{22}$$

$$C_i = [\ \gamma + \delta - 2\gamma\delta\beta + \gamma\delta\beta\alpha^i(\alpha+1)/\alpha\ ]\ S\alpha^i\beta(\alpha-1)/\alpha \tag{23}$$

The total area contribution equation (23) shows that C_i is polynomial in α^i with positive coefficients. Therefore, C_i monotonically increases as i increases, which shows that the higher level wires contribute more to the area than the lower level wires. The lower level power routing schemes are not as important as the higher level schemes. In Mead and Rem [4], it was claimed that only the highest level power wires need be considered when approximating the area of the total array. To check this, we compute Δ, the ratio of the highest level wire contribution plus processing element area to the total area:

$$\Delta = (C_n+S)/A_n \tag{24}$$

$$\Delta = [\ 1 + S^{1/2}\beta(\gamma+\delta)(\alpha-1)/\alpha + S\beta^2\gamma\delta(\alpha^2-1)/\alpha^2\ ] / [\ 1 - \gamma\beta - \delta\beta + \gamma\delta\beta^2 + S^{1/2}\beta(\gamma+\delta-2\gamma\delta\beta) + S\beta^2\gamma\delta\ ] \tag{25}$$

$$\Delta \approx [\ 1 + W(\gamma+\delta)(\alpha-1)/\alpha + W^2\gamma\delta(\alpha^2-1)/\alpha^2\ ] / [\ 1 + W(\gamma+\delta) + W^2\gamma\delta\ ] \tag{26}$$

Equation (26) is graphed in figure 6 for some representative values of W with $\alpha=2$. It can be seen that $\Delta\rightarrow1$ as $W\rightarrow0$ and that $\Delta\rightarrow3/4$ as $W\rightarrow\infty$. When W is small, most of the area comes from the processing elements, so the wire contributions are negligible. When W is large, most of the area comes from the upper level wires. It is ironical that Δ has its minimum value right in the range where we would like W to be. As $\alpha\rightarrow\infty$, $\Delta\rightarrow1$.

Another consideration about the power wire sizes occurs in the lower level modules. In the introduction it was claimed that the minimum wire size β is equal to the processing element current I_{cell} times a design rule ω relating wire width to current capacity. There is another design rule which limits the minimum wire size to β_0 because of fabrication constraints. Thus if $\beta_0>\beta$, the lowest level modules will be wider than equation (10) predicts. In this case, we construct the bottom level module with an α_0, which may have a different value than the α used in the higher level modules. If we are using the staggered tree or staggered comb power routing scheme, the

lowest level module can not be of the type shown in figures 3 and 4, but may be of the form shown in figures 7 and 8. The X_0 and Y_0 values for these bottom level modules are listed in the following table:

	Stree	Scomb
α_0	$2\beta_0/\beta$	β_0/β
X_0	$\alpha_0(1 + (3\alpha_0-1)\beta/2)$	$\alpha_0(1 + 2\alpha_0\beta)$
Y_0	$\alpha_0(1 + 2(\alpha_0-1)\beta)$	$\alpha_0(1 + (\alpha_0-1)\beta)$

The height and width of all modules can be determined from equations (5) and (10) by replacing the 1 with Y_0 and X_0, respectively, replacing β with β_0, and setting the total number of processors S equal to $\alpha_0^2\alpha^{2n}$.

Signal Path Lengths

If the array of processors is to be a logical array of processors as well as physical, each of the processing elements must have communication paths to neighboring processing elements. The cycle time of the array may depend on the time required to transmit data across the communication paths, which in turn is a function of the length of the data path. When all processors are working synchronously, the cycle time is a function of the communication time over the *longest* signal path.

We can determine the longest signal path length, Λ, for each of the power routing schemes by inspection of figures 1-4. For both the staggered and non-staggered tree structures, the longest signal path will have to cross the central VDD line of the top module, since this is the widest gap in the array. For the comb structures, the largest gap is across the vertical pairs of power lines in the top module. In the staggered comb case, the longest signal path will run diagonally, while in the other three cases, the longest path is either strictly horizontal or strictly vertical.

For the tree case, we will define λ_{tree_i} to be the distance from the lowest processing element in a module of level i to the bottom of the module. Λ_{tree} will then be twice $\lambda_{tree_{n-1}}$ plus the width of the top level VDD line. λ_{tree} has the following recurrence relation:

$$\lambda_{tree_0} = 0$$

$$\lambda_{tree_i} = \lambda_{i-1} + (\alpha-1)\beta\alpha^{2i}/2\alpha \quad (27)$$

$$\lambda_{tree_i} = \alpha\beta(\alpha^{2i}-1)/2(\alpha+1) \quad (28)$$

Using (28), we can solve for Λ_{tree}:

$$\Lambda_{tree} = \alpha\beta(\alpha^{2n-2}-1)/2(\alpha+1) + \beta\alpha^{2n} \quad (29)$$

$$\Lambda_{tree} \approx (\alpha^2+\alpha+1)\beta S/\alpha(\alpha+1) \quad (30)$$

Λ_{tree} will have its largest value when $\alpha=2$, which gives an approximate value of $7\beta S/6$. The largest value α can have is $S^{1/2}$, which gives a Λ_{tree} value approaching βS. Λ values for all four power routing schemes are listed in the following table:

	Tree	Comb	Stree	Scomb
Λ	$\frac{(\alpha^2+\alpha+1)\beta S}{\alpha(\alpha+1)}$	$\frac{(\alpha^2-\alpha+4)\beta S}{2\alpha(\alpha-1)}$	$\frac{\alpha\beta S}{\alpha+1}$	$\frac{\beta S\sqrt{[(\alpha^3+\alpha^2-1)^2+\alpha^6]}}{\alpha^3(\alpha+1)}$
$\Lambda\|_{\alpha=2}$	$7\beta S/6$	$\beta S/2$	$2\beta S/3$	$\approx.340\beta S$
$\Lambda\|_{\alpha\to\sqrt{S}}$	βS	$2\beta\sqrt{S}$	βS	$\sqrt{2}\beta\sqrt{S}$

The length of the longest signal path is shortest in the Staggered Comb array. One interesting result from the above analysis is that when a power/speed design decision is made favoring low power over high speed, the resulting array will not only use less power, *but it may run faster!*

Non-Square Processing Elements

If the processing elements are reasonably square, the orientation of the processing elements will not greatly affect the area. If the processing elements are rectangular but definitely not square, a vertical orientation of the longer dimension can have a substantially different area than a horizontal orientation of the longer dimension.

If we normalize the dimensions of the rectangular processing element to $1x\varepsilon$, where $\varepsilon>1$, we can alter equations (5) and (10) to reflect the fact that the processing element is not square. If the longer dimension is oriented vertically, the 1 in equation (5) should be replaced with ε, while if the longer dimension runs horizontally, the modification is made to equation (10).

If $\gamma>\delta$, a direct comparison of the areas resulting from vertical and horizontal orientation shows that the vertical orientation has the smaller area. Conversely, if $\gamma<\delta$, horizontally orienting the longer dimension results in a smaller area. In the tree case, $\gamma>\delta$, so the longer dimension should be vertical, while in the stree case, $\delta>\gamma$, so horizontal orientation is prefered. In both comb cases, $\gamma=\delta$, so either orientation may be used.

Other Geometries

In addition to square and rectangular processing elements, there may be designs using hexagonal or triangular processing elements. Hexagonal arrays have six nearest neighbor communication paths for each processing element, while triangular arrays have three and square arrays have four. Mathematical algorithms for hexagonal arrays using the six communications paths have already been explored [2].

The same techniques used for analyzing square processing elements can be used to analyze hexagonal and triangular processing element arrays.

Hexagonal Processing Elements

Figure 9 shows the layout of one possible hexagonal module. The dimension used to measure the size of a module will be the radius of the inscribed circle. We will define α to be the number of submodules along one side of the hexagonal arrangement of submodules. The number of submodules per module in the hexagonal array is not α^2, as it is in square and triangular arrays, but rather

$$N = 3\alpha^2 - 3\alpha + 1 \tag{31}$$

We can find the inscribed circle's radius by solving a recurrance equation similar to (2), with the following result:

$$Y_i = ((3\alpha-1)/2)^i[\ 1 + (\alpha-1)\beta(9\alpha-6+2\sqrt{3})(2N/(3\alpha-1))^i-1)/6(N-(3\alpha-1)/2)] \tag{32}$$

Using the following definitions, we see that equation (32) is of the same form as equation (5):

$$\rho = (3\alpha-1)/2 \tag{33}$$
$$\sigma = N/\rho \tag{34}$$
$$\gamma = (\alpha-1)(9\alpha-6+2\sqrt{3})/6(N-\rho) \tag{35}$$
$$Y_i = \rho^i\ [\ 1 + \gamma\beta(\sigma^i-1)] \tag{36}$$
$$Y_i \approx \rho^i\ [\ 1 + \gamma\beta\sigma^i\ ] \tag{37}$$

By substituting $\alpha=2$ and the limit $\alpha\rightarrow\infty$ into (33)-(35), we can determine bounds for Y_i:

	$\alpha=2$	$\alpha\rightarrow\infty$
ρ	5/2	$3\alpha/2$
σ	14/5	2α
γ	$(12+2\sqrt{3})/27 \approx .57274$	1/2

The area of a module is found by the following formula:

$$A_i = 2\sqrt{3}\ Y_i^2 \tag{38}$$

The ratio of total area to processing element area is found by (19), with $\delta=\gamma$. This is plotted in figure 10. The wire contribution approximation, Δ, is found in a manner similar to the derivation of (26), with the following result:

$$\Delta = [\ 1 - 1/\rho^{2n} + (2\gamma(1-1/n\rho)+1/\beta\rho^n)W + \gamma^2(N^2-1)W^2/N^2\ ]/R \tag{39}$$

Equation (39) is plotted in figure 11. It can be seen that the approximation is very bad for $\alpha=2$. This is due to the fact that with the hexagonal modules, there is some "wasted" area, or *air*, that is not proportional to the power line widths. The air in the square arrays and in the triangular array vanishes as the power line widths vanish. Not so with the hexagon, as the graph in figure 11 clearly shows.

Triangular Processing Elements

The processing element for triangular arrays is shown in figure 12. The height of a module of level i is Y_i, and the width of the base is ηY_i. The two lower interior angles have measures of θ and ζ. The following relationship between θ, ζ, and η is easily proved using geometric arguments:

$$\eta = \sin(\theta + \zeta)/\sin\theta\sin\zeta \tag{40}$$

We can arrange α^2 of these processing elements as shown in figure 13 to produce a module that is geometrically similar in shape to the original processing element. Since the module has the same shape as the submodule, this arrangement can be used to hierarchically organize a triangular array. One this to notice in figure 13 is that the VDD and Ground lines switch polarity in all "upside down" submodules. This causes no great trouble, except that there must be two types of processing element: One that receives VDD from the apex and Ground from the base, and another with the opposite polarity.

The height of the module comes from three components: the height of the submodules, some of the ground lines, and the VDD line which enters the module at an angle of θ. In some cases, there may be a fourth component which is added to the height when the corner of the "highest upside down" submodule "sticks out too far," as shown in figure 14. If the distance L that the corner protrudes into the VDD line is larger that the width of the power line supplying two submodules, an extra amount must be added tto Y_i. If W is the width of the power line needed to supply one submodule, we can find L by the following equation:

$$L=W\sin\theta/\sin\zeta \tag{41}$$

To insure that the submodule corner does not protrude too far, L must be less than twice W, or

$$\sin\theta\leq 2\sin\zeta \tag{42}$$

Assuming that (41) holds, we can determine the height of a module that includes only three components:

$$Y_i = \alpha Y_{i-1} + (\alpha^2-\alpha)\beta\alpha^{2i-2} + (\alpha^2-1)\beta\alpha^{2i-2}/\cos\theta \tag{43}$$

Solving this recurrance relation gives a value for Y_i of the form of equation (5) with the following value for γ:

$$\gamma = \alpha/(\alpha+1) + 1/\cos\theta \tag{44}$$

Values for γ with $\alpha=2$ and as $\alpha\rightarrow\infty$ are shown in figure 15. The following equation gives the area of a module:

$$A_i = \eta Y_i^2/2 \tag{45}$$

The ratio of total area to processing element area is found by (19), with $\delta=\gamma$, and is plotted in figure 16 for values of $\theta=0°$, 45°, and 60°. Δ can be found by equation (26), again with $\delta=\gamma$. Δ is graphed in figure 17 for same values of θ as figure 16.

Conclusions

We have seen several schemes for routing power to an array of processing elements, and have developed techniques for comparing those schemes. With this modest background, design tradeoffs for array and tree machines can be made during the initial design stages.

Two of the more interesting results found by this paper concern design tradeoffs considering power consumption:

> In making a power vs. area design decision, choosing the low power design may save not only power, *but also area*!
>
> In making a power vs. speed design decision, choosing the low power design may save not only power, *but also increase the speed*!

Appendix

To help emphasize the fact that the above analysis can be used to produce real results, figures 18-21 show ICL [1] implementations for each of the square processing element power routing schemes. The code for each of these figures is also listed. The code shown is almost identically the code that would be used in actual designs.

References

[1] Ayres, Ronald
"ICL Reference Manual"
PhD. Thesis (California Institute of Technology, 1978)

[2] Kung, H. T., Charles E. Leiserson
"Algorithms for VLSI Processor Arrays"
section 8.2 of
Mead, Carver A., Lynn Conway
Introduction to VLSI Systems
in preparation.

[3] Mead, Carver A., Lynn Conway
"Introduction to VLSI Systems"
in preparation.

[4] Mead, Carver A., Martin Rem
"Cost and Performance of VLSI Computing Structures"
Display file #n, California Institute of Technology, 1978

Branching Ratio = 4

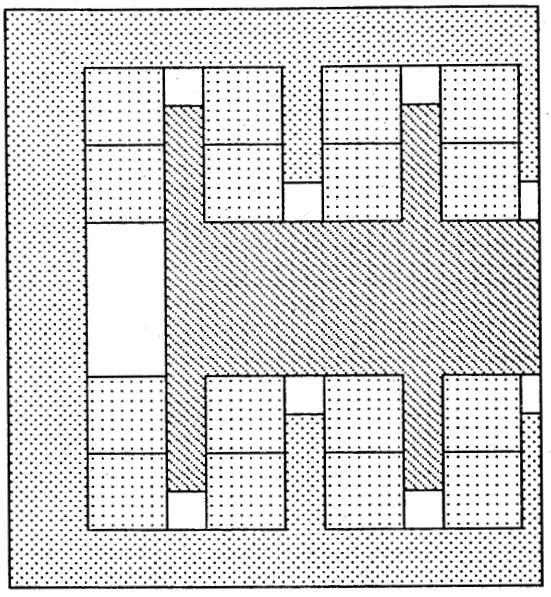


Figure 1. Tree Structure Power Routing.

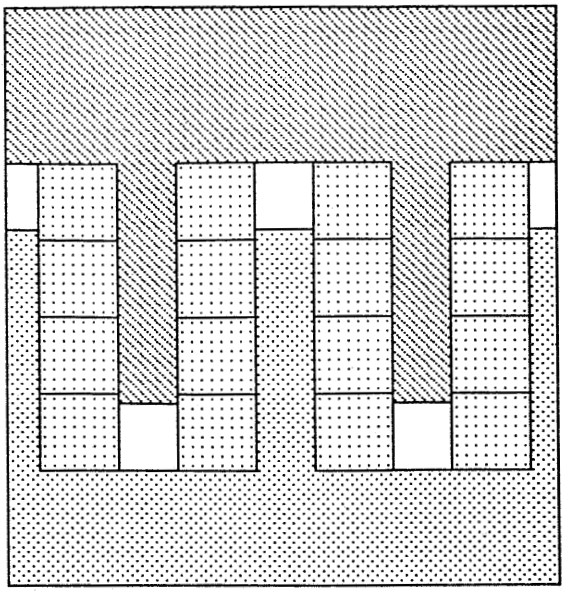


Figure 2. Comb Structure Power Routing.

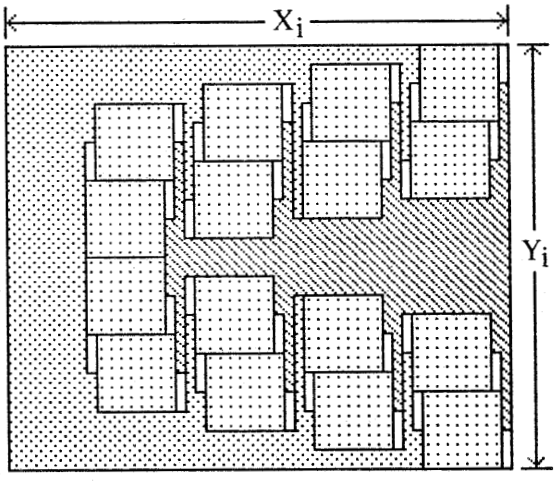


Figure 3. Staggered Tree Power Routing.

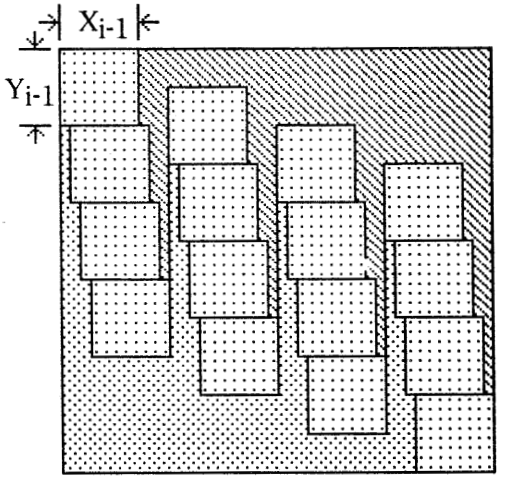


Figure 4. Staggered Comb Power Routing.

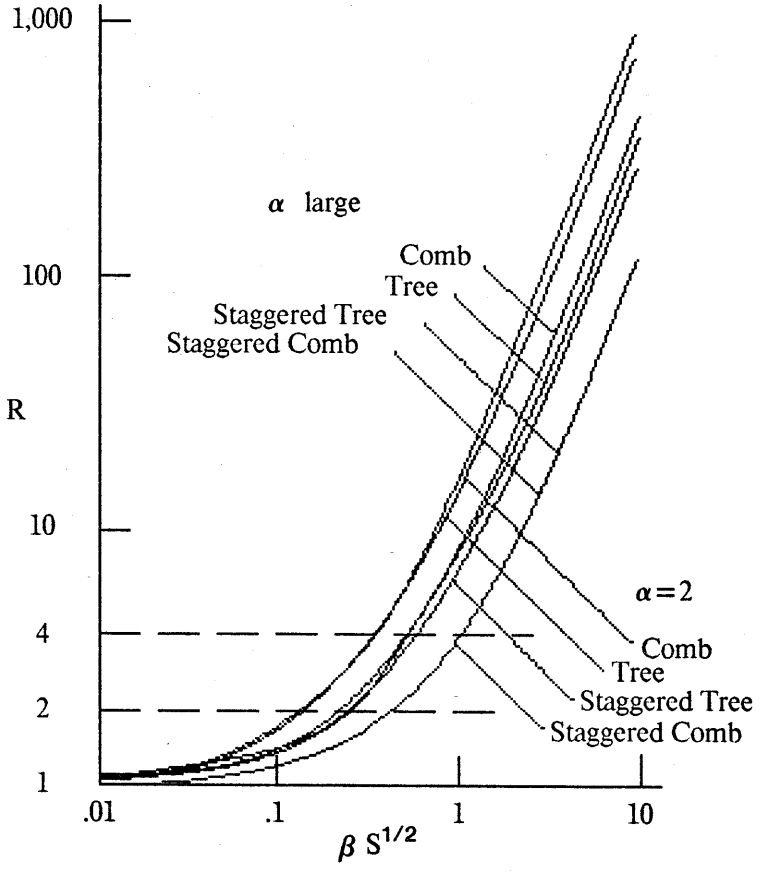


Figure 5. Ratio of Total Area to Processor Area

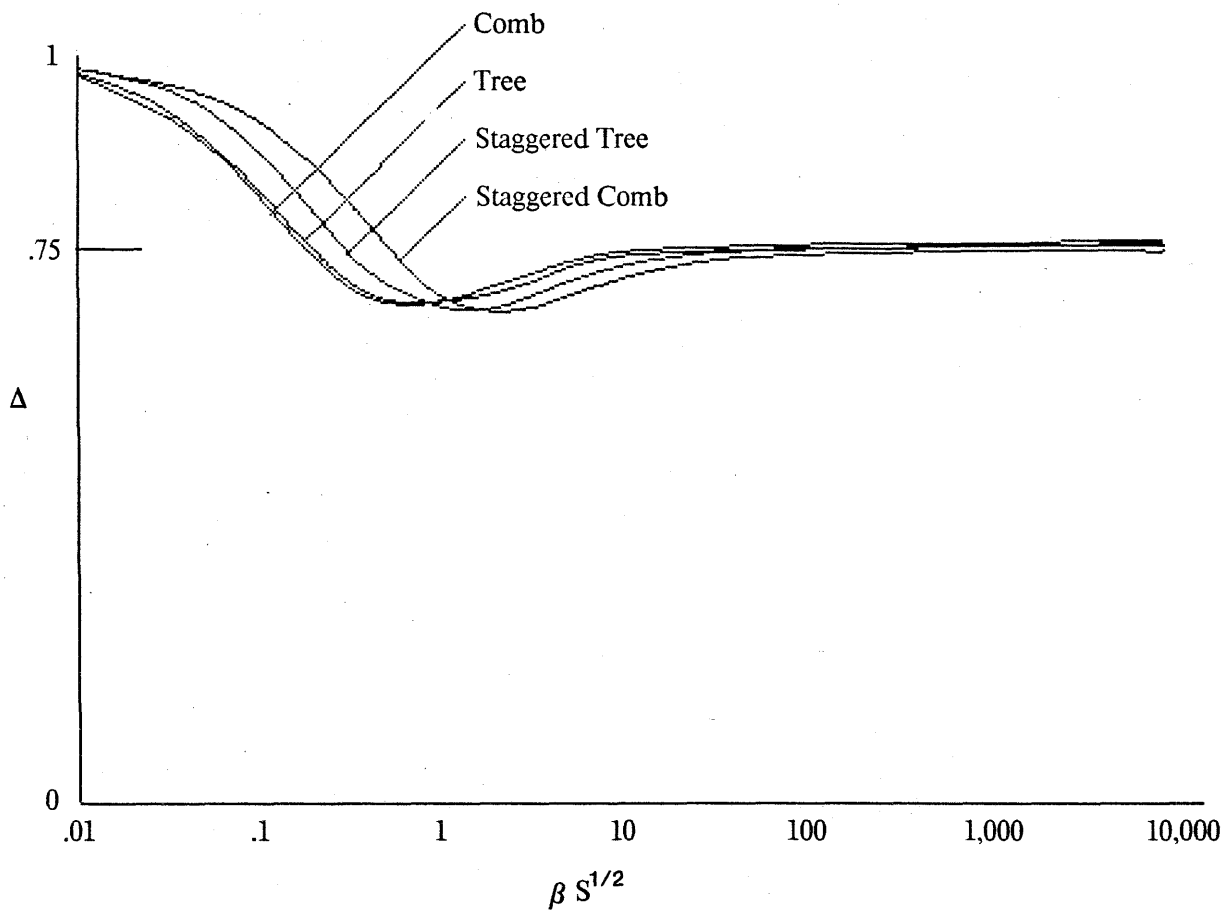


Figure 6. Wire Contribution Approximation for Square Processing Elements
Branching Ratio = 2

Branching Ratio = 4

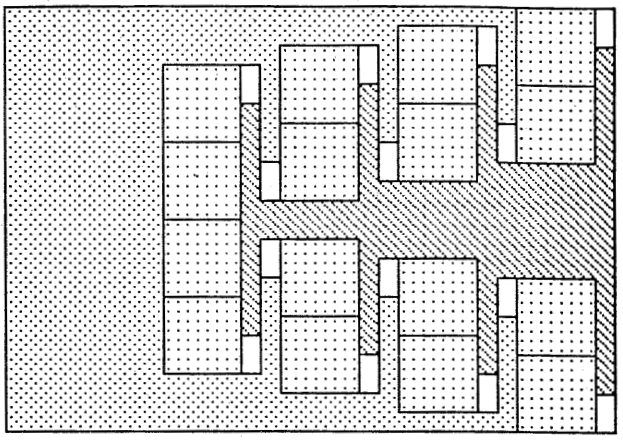


Figure 7. Level 1 Staggered Tree Module.

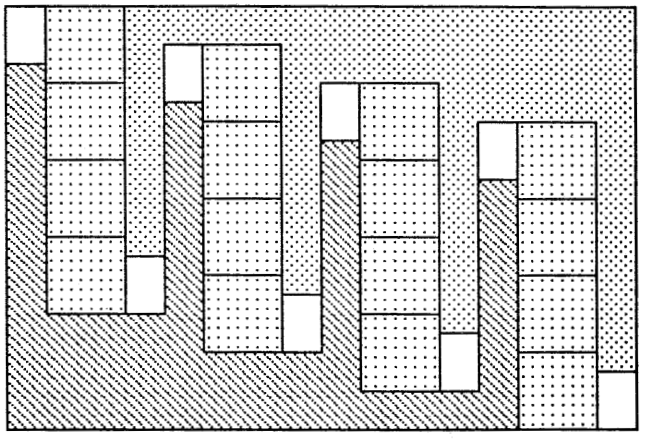


Figure 8. Level 1 Staggered Comb Module.

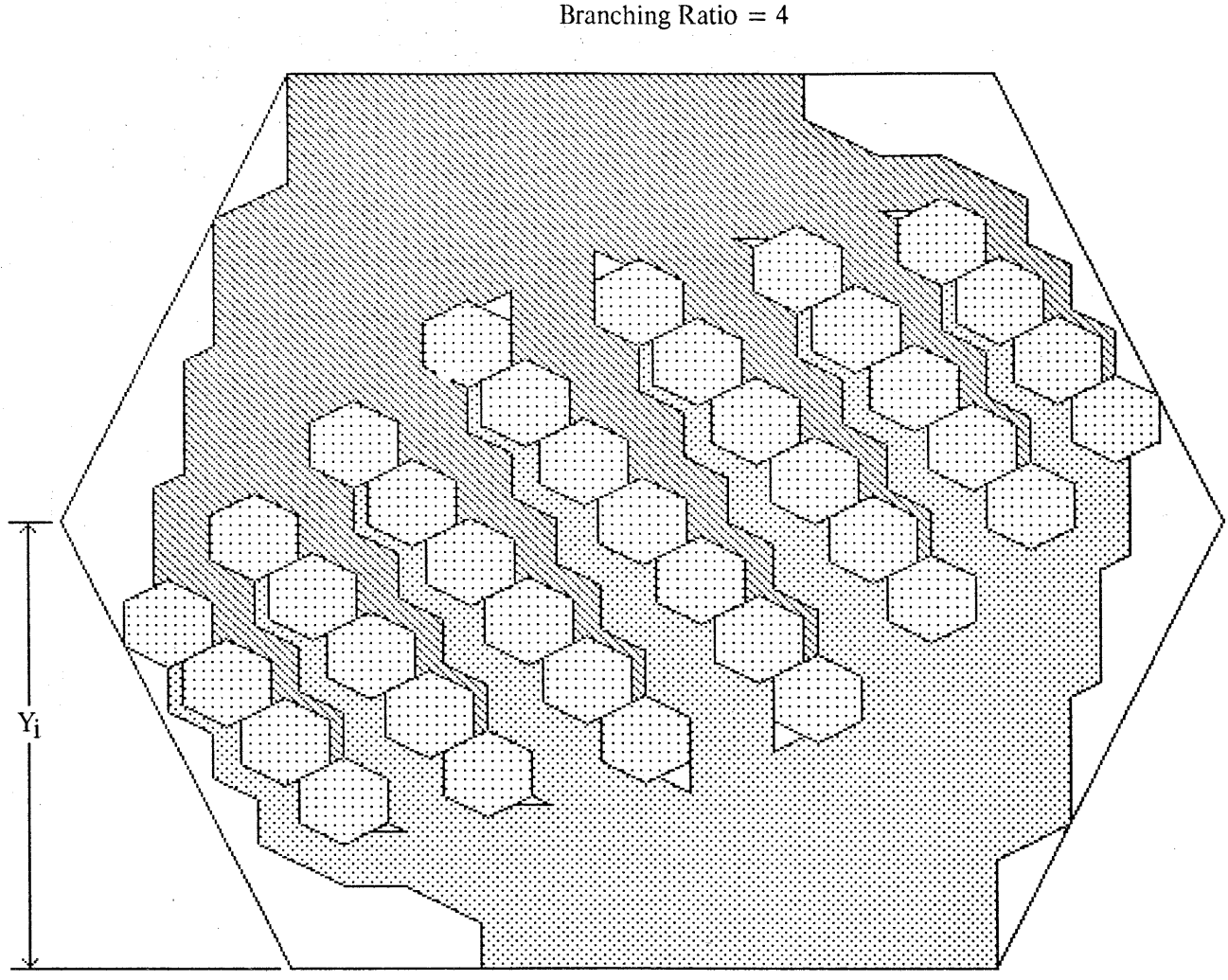


Figure 9. Power Routing Scheme for Hexagonal Processing Elements.

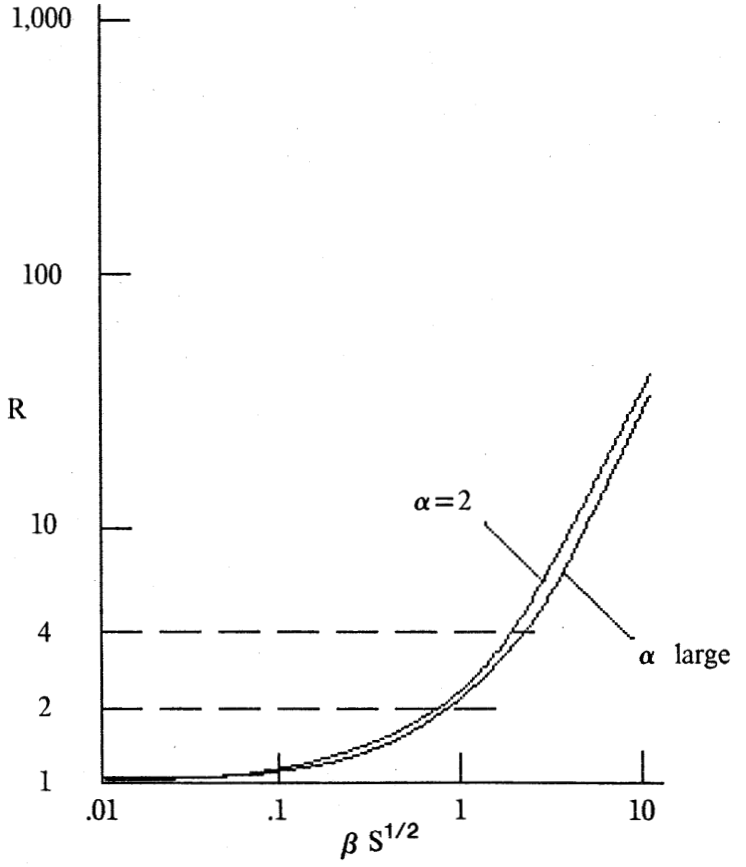


Figure 10. Ratio of Total Area to Processor Area (Hex).

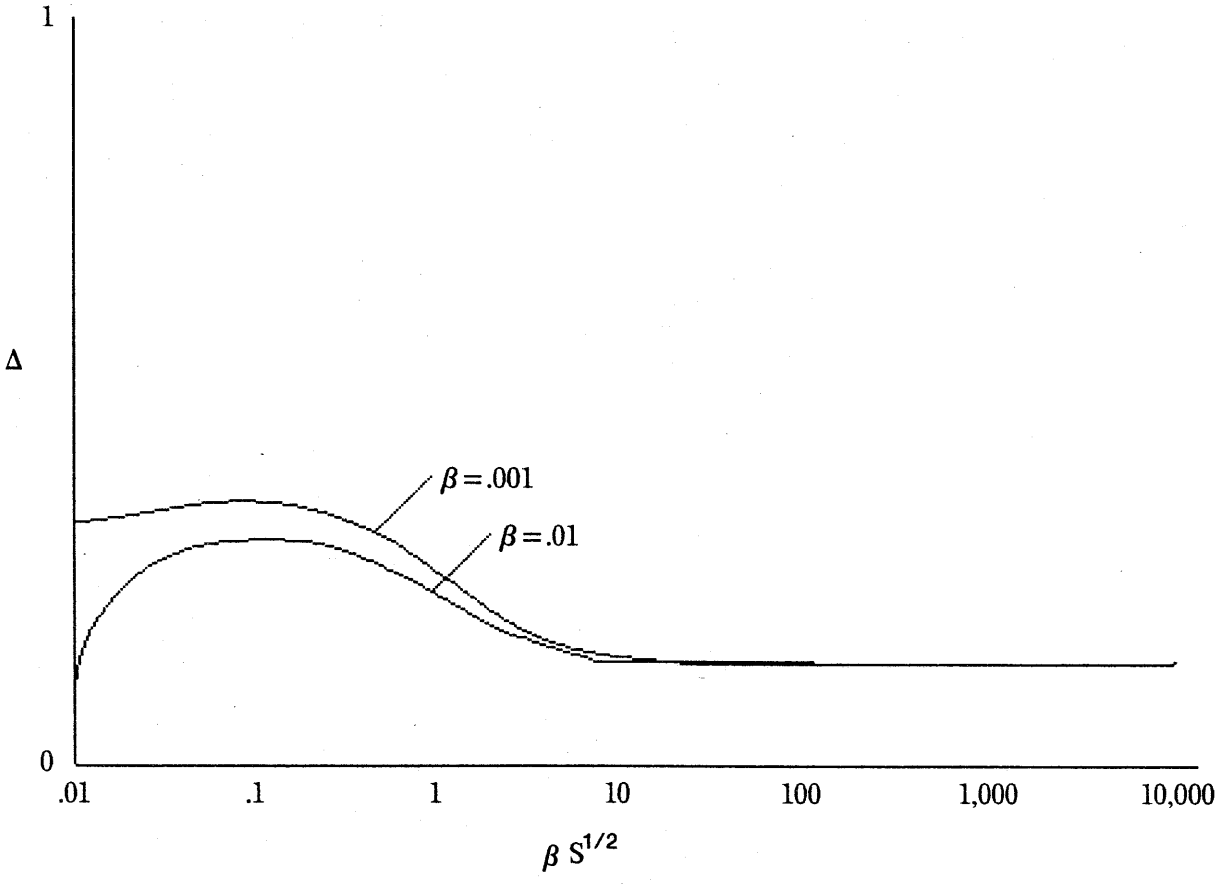


Figure 11. Wire Contribution Approximation (Hex).
Branching Ratio = 2

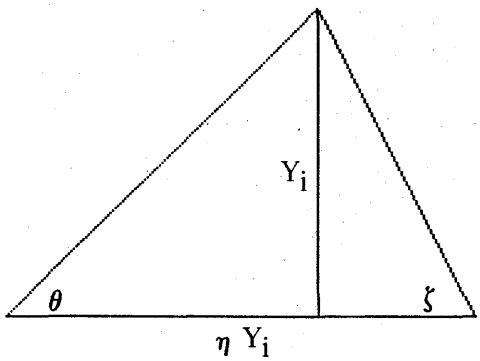


Figure 12. Triangular Processing Element.

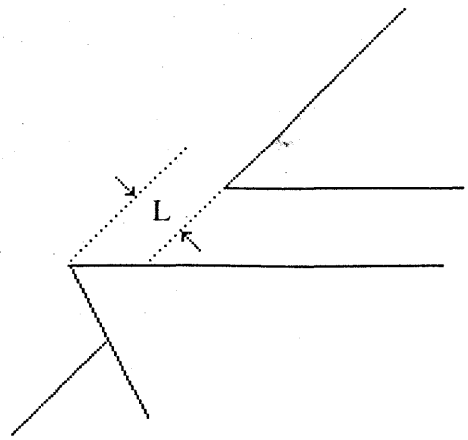


Figure 14. Source of Possible Fourth Height Component.

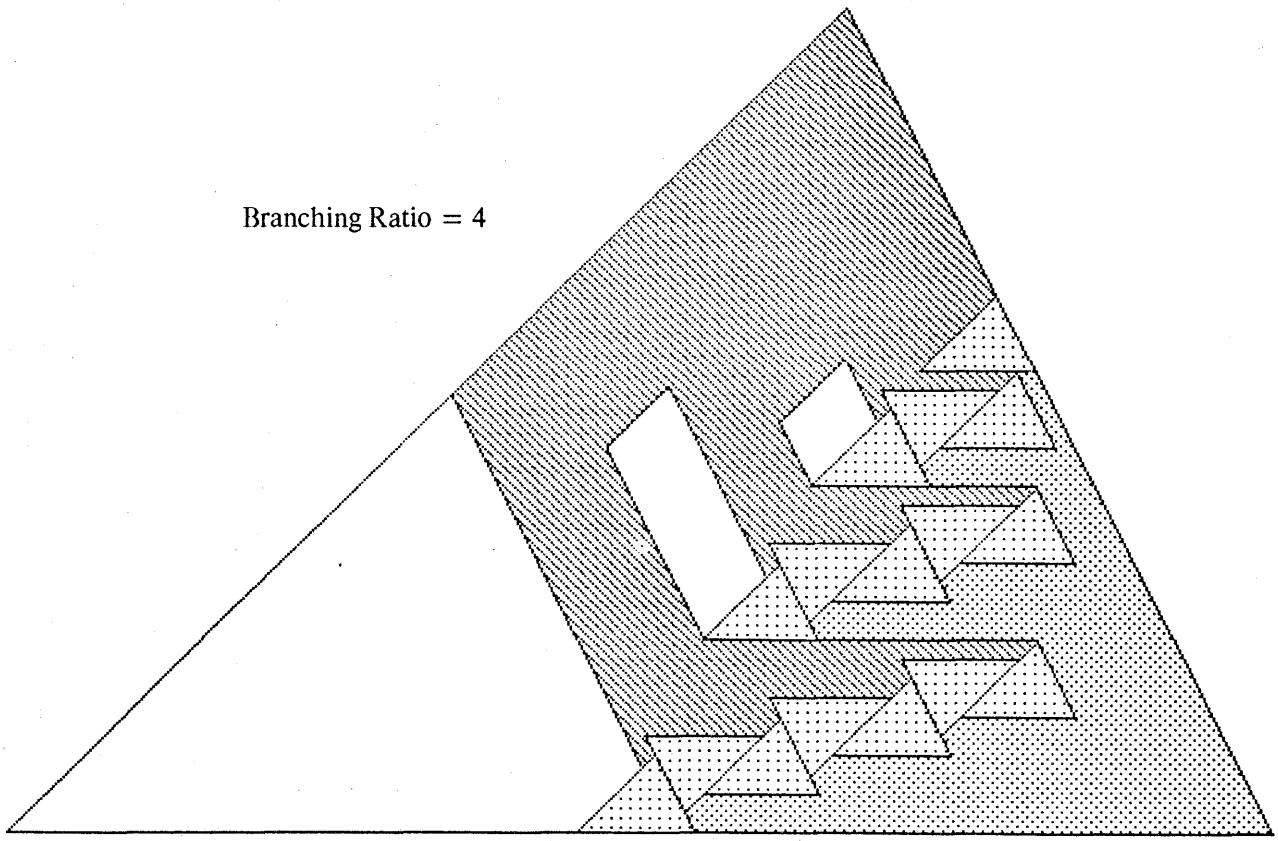


Figure 13. Layout of a Triangular Array Module.

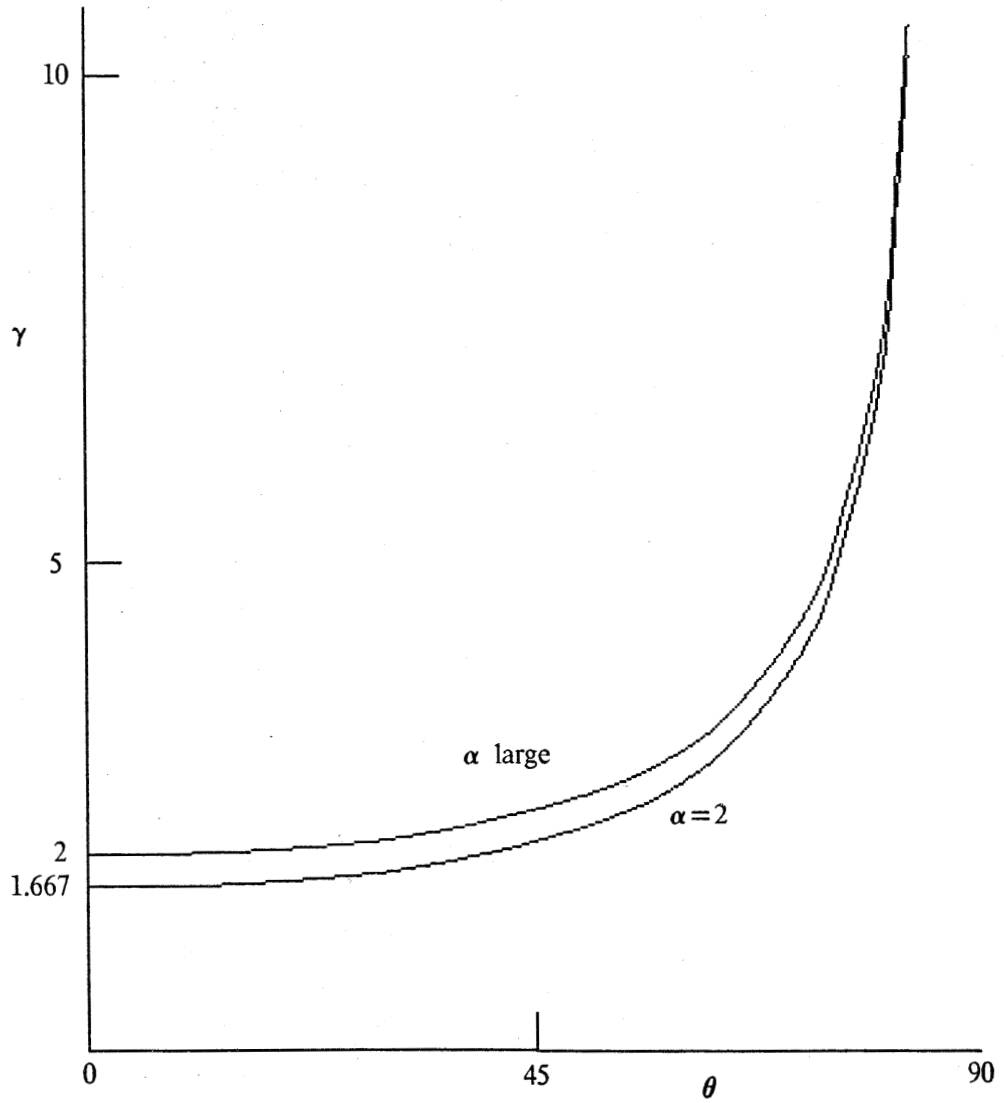


Figure 15. γas a Function of α,θ (Triangular).

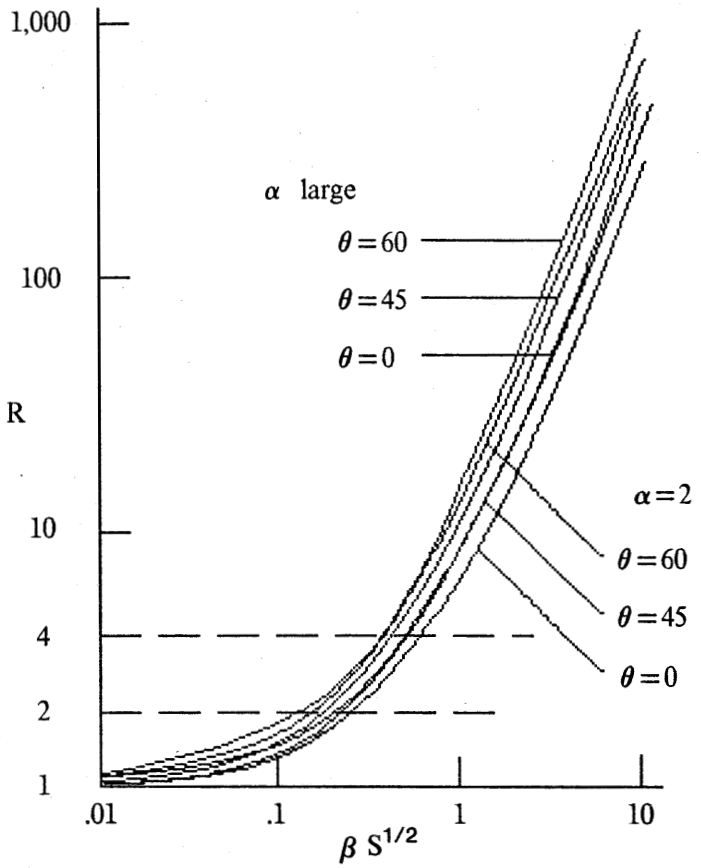


Figure 16. Ratio of Total Area to Processor Area (Triangular).

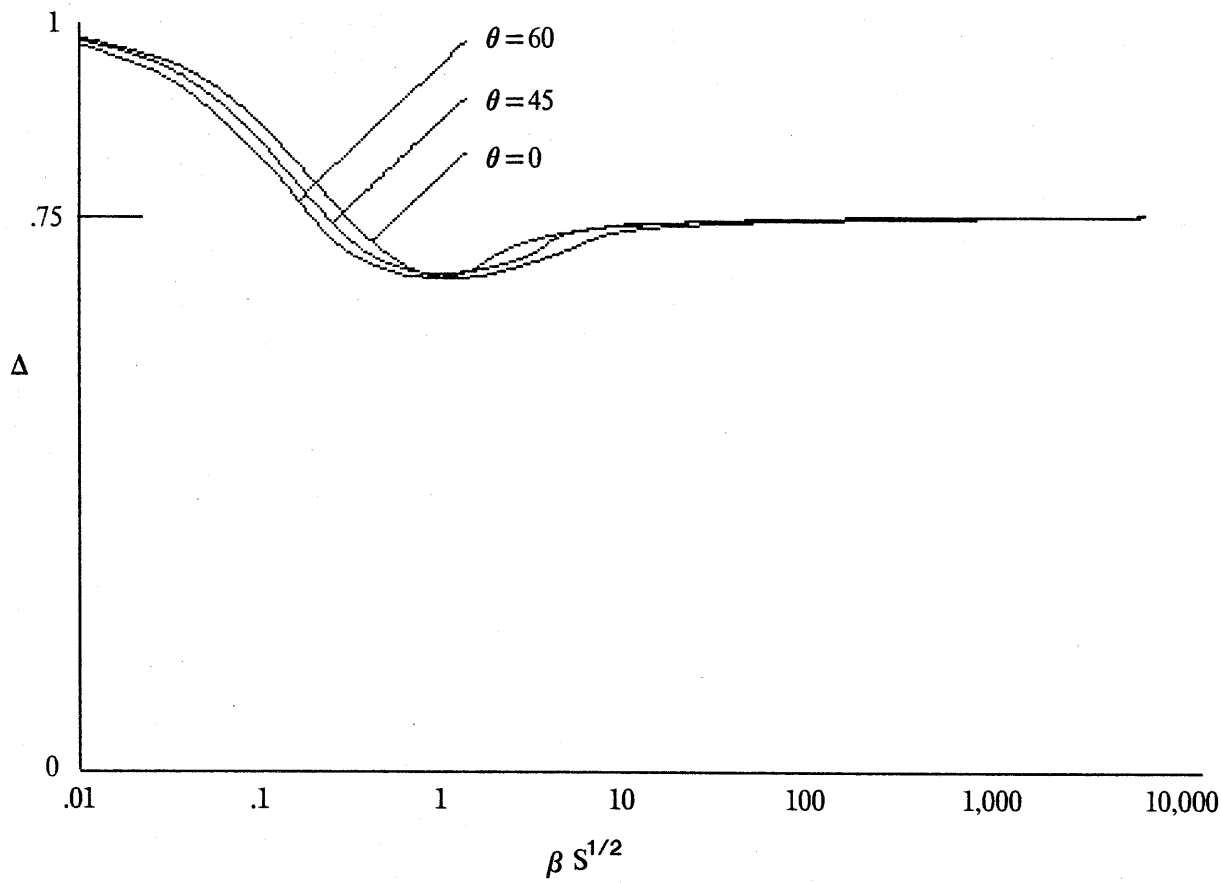
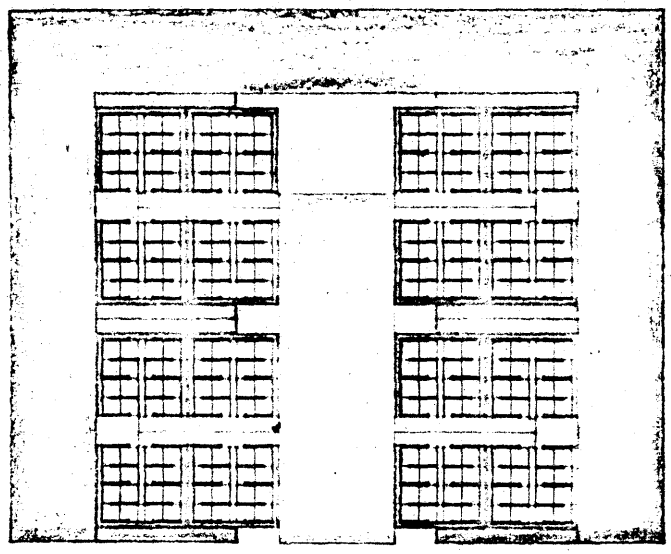
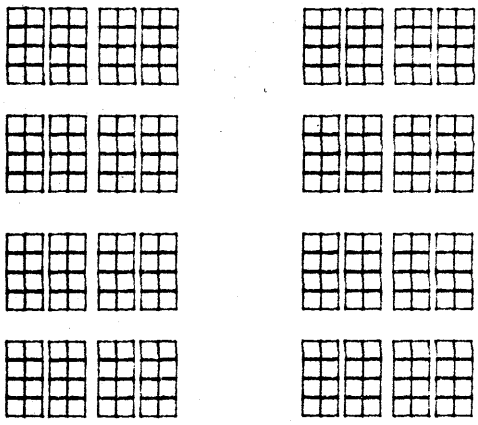


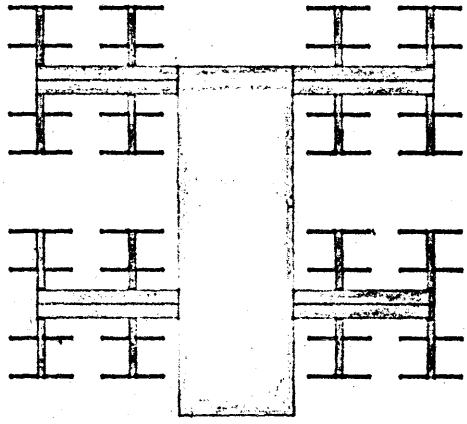
Figure 17. Wire Contribution Approximation (Triangular).
Branching Ratio = 2



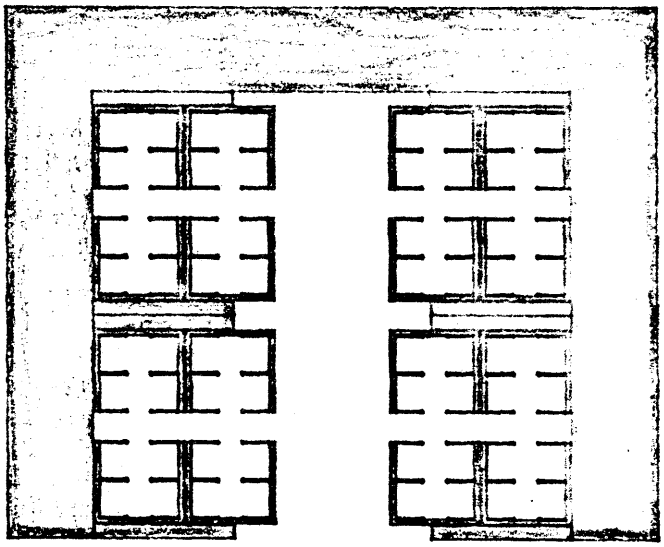
Composite Drawing



Processing Element Layout



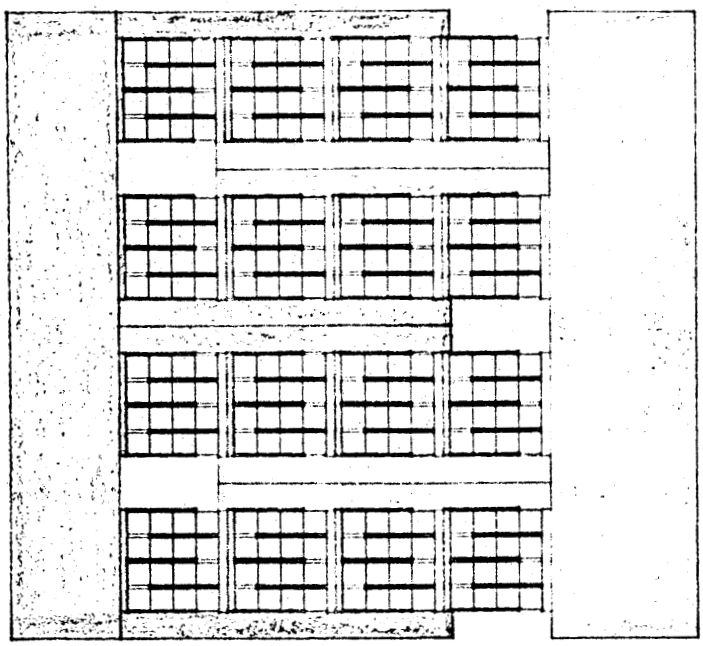
VDD Net



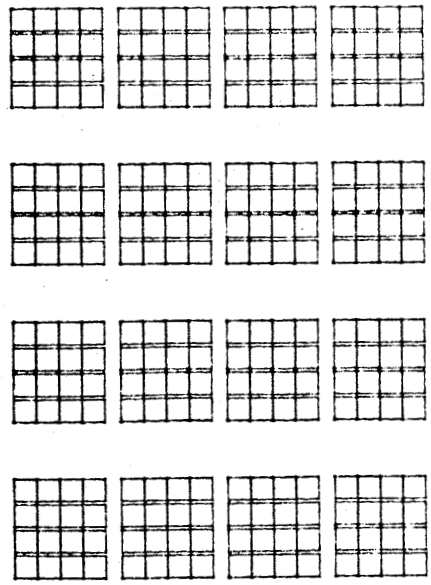
Ground Net

Figure 18. An ICL Implementation of the Tree Structure Power Routing Method.
(The ICL Listing is on the Following Page).

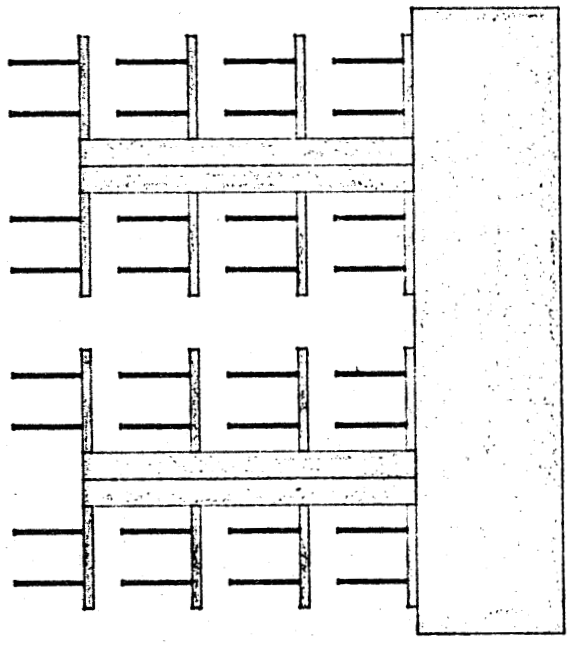
```
VAR ALPH,ALPH2,HALPH=INT; SIZE,SZ2,SZ3,SZ4,A,B,C,D,E,F,BETA=REAL;
VAR S=SYMBOL;
BETA:=.001;
ALPH:=2;

DEFINE PS:
    S:=GB(0#0,1#1);
    ALPH2:=ALPH*ALPH;
    HALPH:=ALPH/2;
    SIZE:=BETA;
ENDDEFN

DEFINE P:
    SZ2:=SIZE*ALPH2;
    SZ3:=SIZE*ALPH*(ALPH-1);
    SZ4:=SZ3/2;
    [LOW:A#B HIGH:C#D]:=S.MBB;
    E:=C-A+SIZE*HALPH;
    F:=(D-B-SIZE)/2;
    S:={S\DISPLACED_BY [IX:0 IY:D-B NX:1 NY:HALPH];
        RB(C#B,C+SIZE*HALPH#B+HALPH*(D-B)-F);
        BB(A-SIZE*HALPH#B+F,A#B+HALPH*(D-B))};
    [LOW:A#B HIGH:C#D]:=S.MBB;
    S:={S;
        S\MIRY\AT 2*C#0};
    S:=S\DISPLACED_BY [IX:2*(C-A) IY:0 NX:HALPH NY:1];
    [LOW:A#B HIGH:C#D]:=S.MBB;
    S:={S;
        S\MIRX\AT 0#2*B-SZ2;
        RB(A+E#B-SZ2,C#B);
        BW(SZ3,{C-SZ4#D+SZ4;A-SZ4#.;.#2*B-SZ2-D-SZ4;C-SZ4#.})};
    SIZE:=SIZE*ALPH2;
ENDDEFN

DEFINE DRAW(N:INT):
    PS;
    DO P; REPEAT N;
    PLOT(S,'TEMP'\AIF {RED;BLU});
    PLOT('TEMP');
ENDDEFN
```



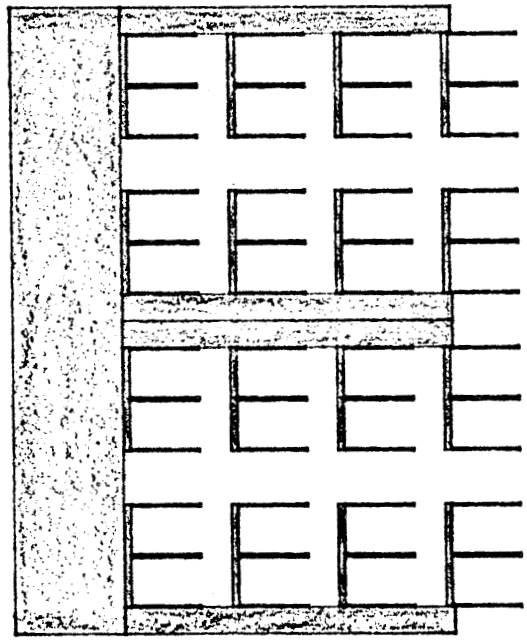
Composite Drawing



Processing Element Layout



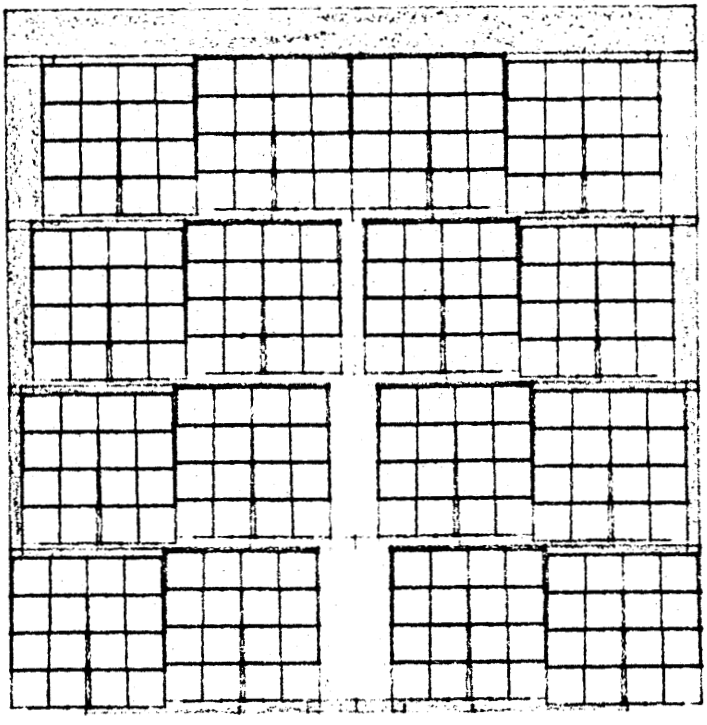
VDD Net



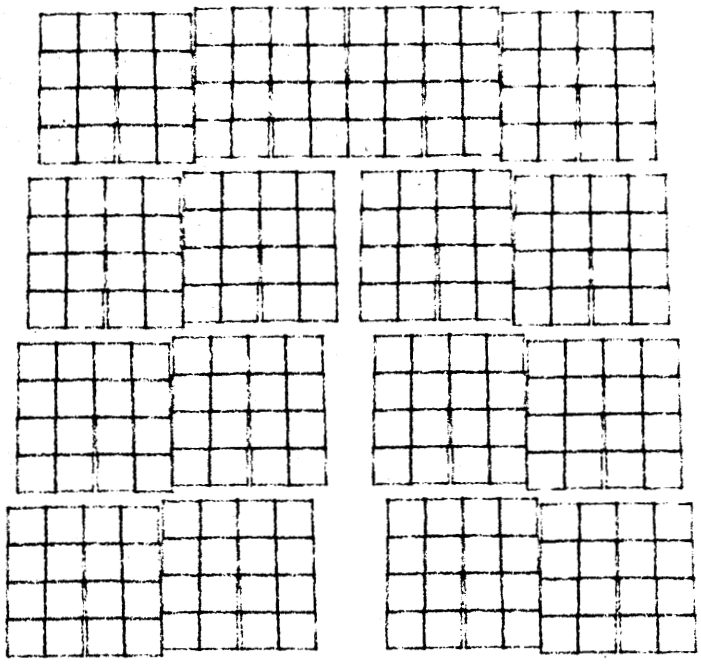
Ground Net

Figure 19. An ICL Implementation of the Comb Structure Power Routing Method.
(The ICL Listing is on the Following Page).

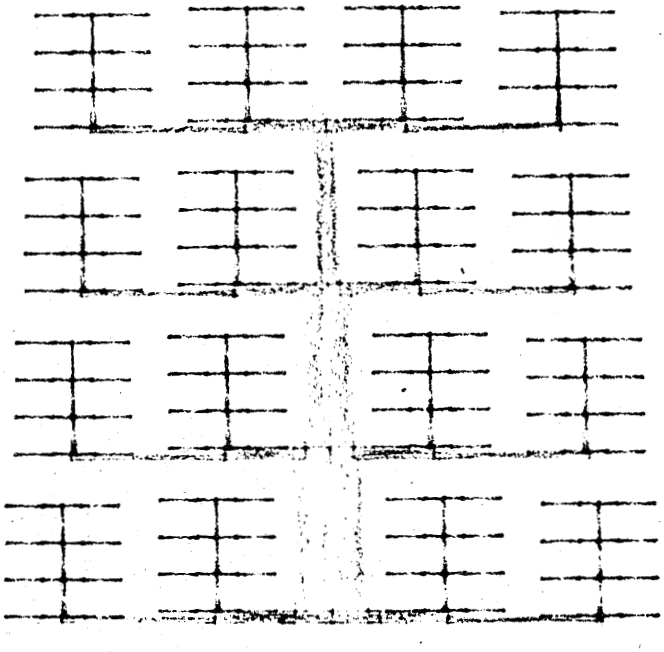
```
VAR ALPH,ALPH2,HALPH=INT; SIZE,SZ2,SZ3,A,B,C,D,E,BETA=REAL;
VAR S=SYMBOL;
BETA:=.001;
ALPH:=2;

DEFINE PS:
    S:=GB(0#0,1#1);
    ALPH2:=ALPH*ALPH;
    HALPH:=ALPH/2;
    SIZE:=BETA;
ENDDEFN

DEFINE P:
    SZ2:=SIZE*ALPH2;
    SZ3:=SIZE*ALPH*(ALPH-1);
    [LOW:A#B HIGH:C#D]:=S.MBB;
    E:=D-B-SIZE;
    S:={S\DISPLACED_BY [IX:0 IY:D-B NX:1 NY:ALPH];
        RB(C#B+E,C+SIZE*(ALPH-1)#B+ALPH*(D-B));
        BB(A-SIZE*(ALPH-1)#B,A#B+ALPH*(D-B)-E)};
    [LOW:A#B HIGH:C#D]:=S.MBB;
    S:={S;
        S\MIRY\AT 2*C#0};
    S:=S\DISPLACED_BY [IX:2*(C-A) IY:0 NX:HALPH NY:1];
    [LOW:A#B HIGH:C#D]:=S.MBB;
    S:={S;
        RB(A#D,C#D+SZ2);
        BB(A#B-SZ3,C#B)};
    SIZE:=SIZE*ALPH2;
ENDDEFN

DEFINE DRAW(N:INT):
    PS;
    DO P; REPEAT N;
    PLOT(S,'TEMP'\AIF {RED;BLU});
    PLOT('TEMP');
ENDDEFN
```



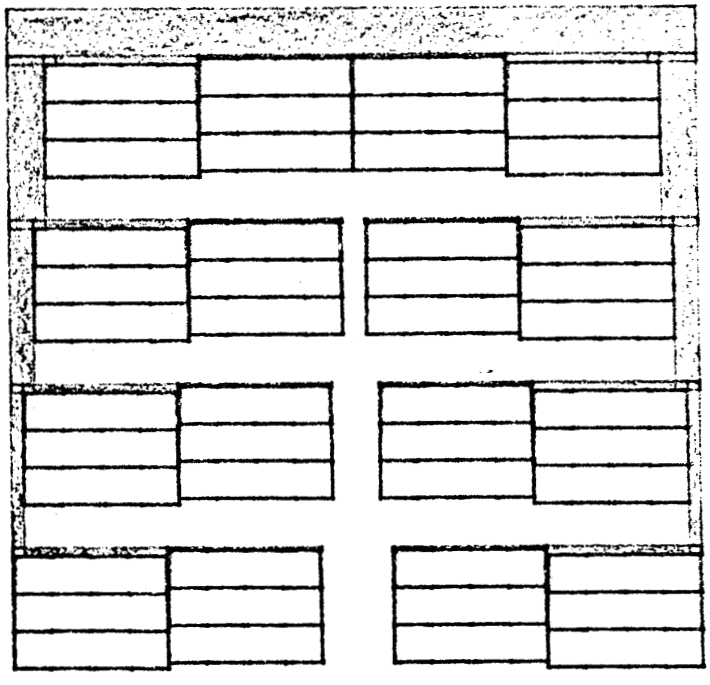
Composite Drawing



Processing Element Layout



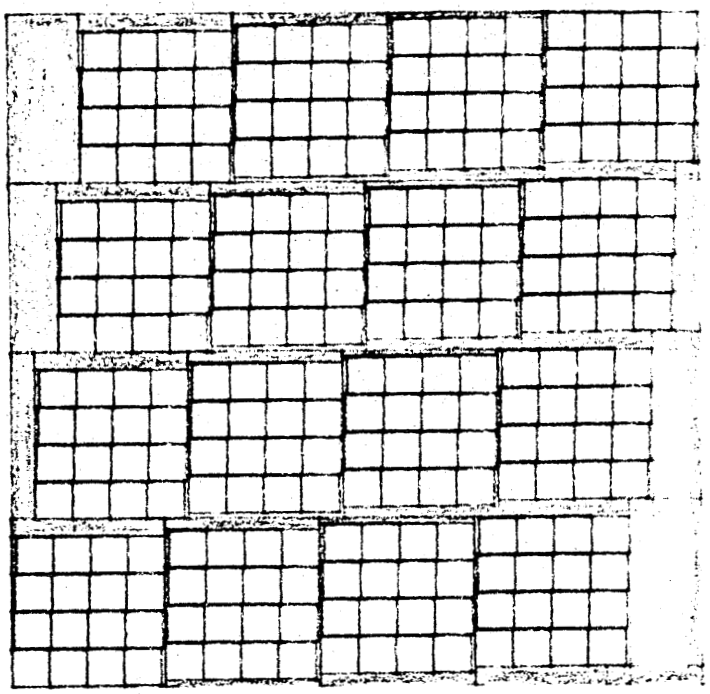
VDD Net



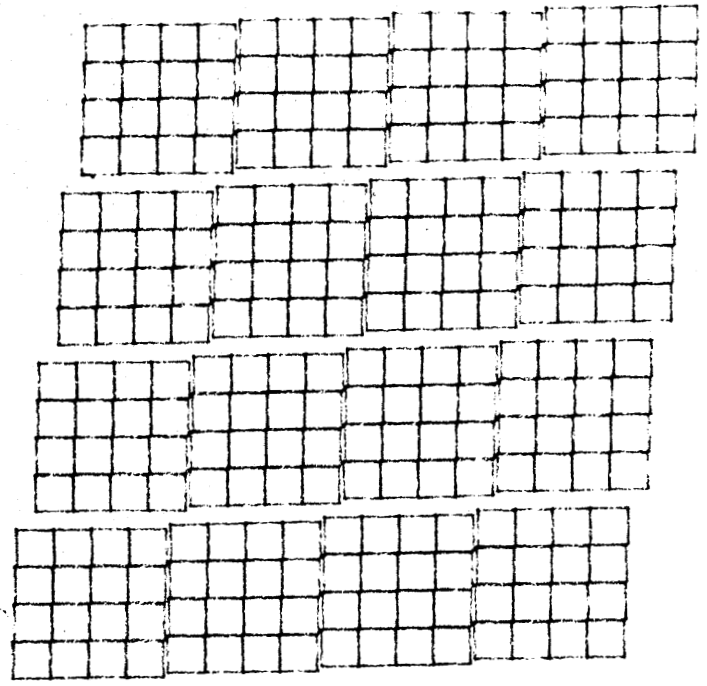
Ground Net

Figure 20. An ICL Implementation of the Staggered Tree Power Routing Structure.
(The ICL Listing is on the Following Page).

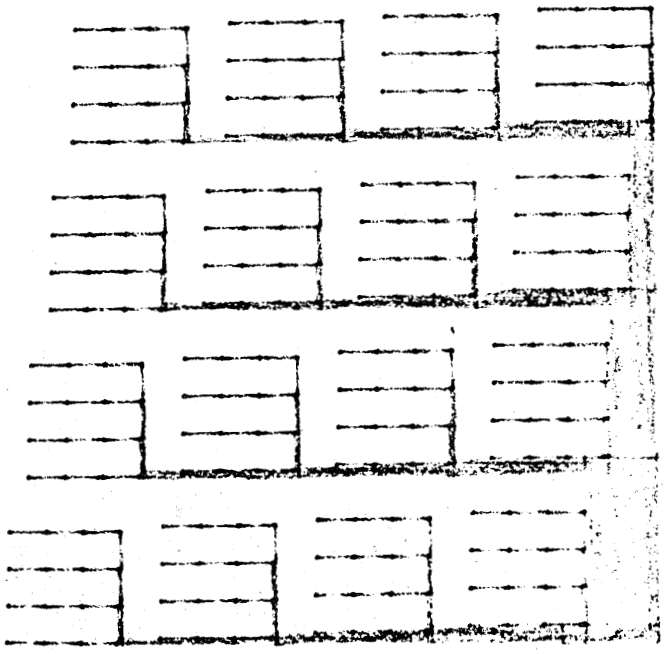
```
VAR ALPH,HALPH,ALPH2,I=INT; BETA,SIZE,SZ2,SZ3,SZ4,SZ5,A,B,C,D=REAL;
VAR S=SYMBOL;
BETA:=.01;
ALPH:=2;

DEFINE PS:
    S:=GB(0#0,1#1);
    SIZE:=BETA;
    HALPH:=ALPH/2;
    ALPH2:=ALPH*ALPH;
ENDDEFN

DEFINE STAG1(I:INT)=SYMBOL:
    BEGIN VAR T=SYMBOL; E,F,G,H,J,X,Y=REAL;
    DO  [LOW:E#F HIGH:G#H]:=S.MBB;
        X:=(H-F-SIZE)/2;
        Y:=(H+F+SIZE)/2;
        J:= IF I=1 THEN F ELSE F-X FI;
        T:={S\AT SZ2*(I-1)#0;
            BB(E-SZ3#F,E+SZ2*(I-1)#H);
            RB(G+SZ2*(I-1)#J,G+ALPH*SZ3#Y)};
    GIVE T\DISPLACED_BY 0#(H-F)*(I-1)
    END
ENDDEFN

DEFINE STAG2(I:INT)=SYMBOL:
    BEGIN VAR T=SYMBOL; E,F,G,H,K,L,X=REAL;
    DO  [LOW:E#F HIGH:G#H]:=S.MBB;
        X:=G-SIZE*HALPH;
        K:= IF I=ALPH THEN G ELSE G-E+X FI;
        L:= IF I=1 THEN E ELSE 2*E-G FI;
        T:={S\AT 0#SZ5*(I-1);
            BB(L#H+SZ5*(I-2),E+SZ4#H+(ALPH-1)*SZ5);
            RB(X#F,K#F+SZ5*I)};
    GIVE T\DISPLACED_BY (G-E)*(I-1)#0
    END
ENDDEFN

DEFINE P:
    SZ2:=SIZE;
    SZ3:=SZ2/2;
    SZ5:=SZ3*ALPH;
    SZ4:=SZ5*(ALPH-1)/ALPH;
    S:={COLLECT STAG1(I) FOR I FROM 1 TO HALPH;};
    S:={COLLECT STAG2(I) FOR I FROM 1 TO ALPH;};
    [LOW:A#B HIGH:C#D]:=S.MBB;
    S:={S;S\MIRX\AT 0#2*B};
    SIZE:=SIZE*ALPH2;
ENDDEFN

DEFINE FIX:
    [LOW:A#B HIGH:C#D]:=S.MBB;
    S:={S;
        BB(A-SIZE/2#B,A#D)};
ENDDEFN

DEFINE DRAW(N:INT):
    PS;DO P; REPEAT N;FIX;
    PLOT(S,'TEMP');
ENDDEFN
```



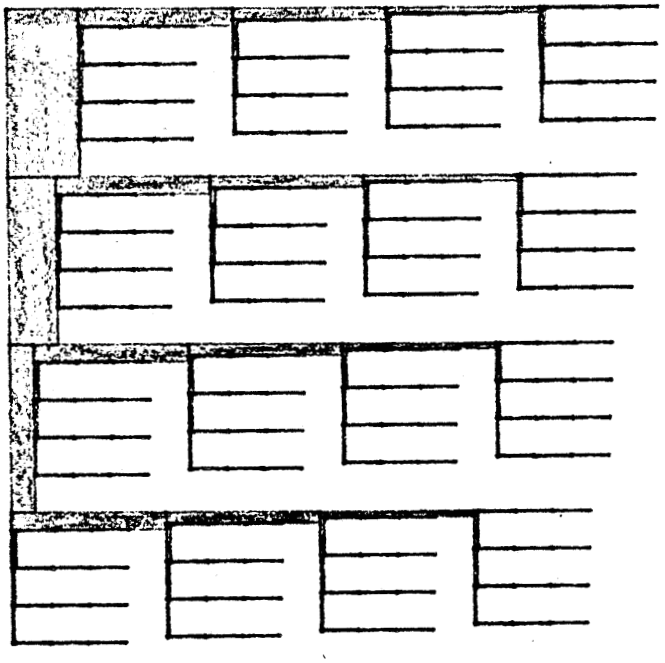
Composite Drawing



Processing Element Layout



VDD Net



Ground Net

Figure 21. An ICL Implementation of the Staggered Comb Power Routing Structure.
(The ICL Listing is on the Following Page).

```
VAR ALPH,HALPH,ALPH2,I=INT; BETA,SIZE,SZ2,SZ3,SZ4,SZ5,A,B,C,D,X,Y=REAL;
VAR S=SYMBOL;
BETA:=.01;
ALPH:=2;

DEFINE PS:
    S:=GB(0#0,1#1);
    SIZE:=BETA;
    HALPH:=ALPH/2;
    ALPH2:=ALPH*ALPH;
ENDDEFN

DEFINE STAG1(I:INT)=SYMBOL:
    BEGIN VAR T=SYMBOL; E,F,G,H=REAL;
    DO  [LOW:E#F HIGH:G#H]:=S.MBB;
        T:= IF I=1 THEN
                {S\AT SZ2*(I-1)#0;
                 RB(G+SZ2*(I-1)#F,G+(ALPH-1)*SZ2#H)}
            EF I=ALPH THEN
                {S\AT SZ2*(I-1)#0;
                 BB(E#F,E+SZ2*(I-1)#H)}
            ELSE
                {S\AT SZ2*(I-1)#0;
                 BB(E#F,E+SZ2*(I-1)#H);
                 RB(G+SZ2*(I-1)#F,G+(ALPH-1)*SZ2#H)} FI;
    GIVE T\DISPLACED_BY 0#(F-H)*(I-1)
    END
ENDDEFN

DEFINE STAG2(I:INT)=SYMBOL:
    BEGIN VAR T=SYMBOL; E,F,G,H,K,L=REAL;
    DO  [LOW:E#F HIGH:G#H]:=S.MBB;
        K:= IF I=ALPH THEN G ELSE G-Y+X FI;
        L:= IF I=ALPH THEN E+SZ2 ELSE G FI;
        T:={S\AT 0#-SZ5*(I-1);
            BB(E#F-SZ5*(ALPH-1),L#F-(I-1)*SZ5);
            RB(X#H-SZ5*I,K#H)};
    GIVE T\DISPLACED_BY (G-E)*(I-1)#0
    END
ENDDEFN

DEFINE P:
    SZ2:=SIZE;
    SZ3:=SZ2/2;
    SZ5:=SZ2*ALPH;
    SZ4:=SZ5*2;
    [LOW:Y#B HIGH:X#D]:=S.MBB;
    S:={COLLECT STAG1(I) FOR I FROM 1 TO ALPH;};
    S:={COLLECT STAG2(I) FOR I FROM 1 TO ALPH;};
    SIZE:=SIZE*ALPH2;
ENDDEFN

DEFINE DRAW(N:INT):
    PS;
    DO P; REPEAT N;
    PLOT(S,'TEMP');
ENDDEFN
```